15. Масиви

Проф. д-р Емил Хаджиколев

1. Масиви.
2. Видове масиви.
3. Декларация и инициализация на масив.
4. Работа с едномерен масив.

Масиви

- **Масивите са сложни обекти, съдържащи множество от елементи**.
- Казва се, че те описват **колекция от елементи**. Всеки елемент се представя като двойка величини „ключ-стойност” (key-value).
- **Ключовете** в един масив **са уникални**, а стойностите може да се повтарят.
- **Чрез името на променливата и стойността на ключа може да се достъпи съответната на ключа стойност**.
- В масиви може да се описва информация за хора, фирми, населени места, измервания и всякакви други обекти от реалния свят.

Видове масиви в ЕП

- **Според типа на ключовете,** масивите могат да бъдат:
  - **Обикновени** – определят се като **последователност от еднотипни елементи.** Ключовете са последователни цели числа, а стойностите на масива са от един и същи тип. **Ключовете** в този случай **се наричат индекси**, като първият индекс има стойност 0, следващият – 1 и т.н.
  - **Асоциативни** – ключовете и стойностите може да са от произволен тип.
- **Според броя на размерностите,** масивите са:
  - **Едномерни** – за достъп до една стойност се използва един ключ;
  - **Двумерни** – за достъп до една стойност се използва комбинация от два ключа. Използват се за представяне на матрици;
  - **Многомерни** – може да се използват за представяне на йерархични структури.
- **За обхождането и обработката на масиви се използват цикли**.

Декларация на статичен масив

- В C++ се използват **обикновени масиви**. Те са последователност от еднотипни елементи. Индексите са последователни цели числа, първият от които е 0, а стойностите са от един и същи тип.
- Масивите се декларират по начин, подобен на другите променливи, като след името на променливата се задава брой на елементите в правоъгълни скоби ([]) :

```
<тип> <име на променлива за масива>[<брой на елементите на масива>];
```

- При декларация на **статични масиви** могат да се задават само **константи за брой на елементите**.
- Този брой трябва да е известен по време на компилиране на програмата (и затова е константа).
- При **динамичните масиви**, които ще разгледаме в следваща лекция, **броят на елементите може да се задава с променлива**, по време на изпълнение на програмата.

Примери за едномерни статични масив

```
int intArray[3];          // масив от цели числа
double doubleArray[4];    // масив от реални числа
string names[5];          // масив с имена - низове
Person persons[6];        // масив с елементи от съставен тип Person
// само константа за цяло число (вкл. литерал, каквито са горните примери) може да се ползва за брой на елементите
const int size = 3;
int arr[size];
```

Декларация на многомерен статичен масив

- Размерностите на масивите се определят от използваните двойки правоъгълни скоби.
- Двумерен масив се записва с две двойки правоъгълни скоби, тримерен – с три и т.н. Например:

```
// двумерен масив – матрица – с два реда и три колони
int matrix[2][3];

double array3[3][4][5];    // тримерен масив
```

Едномерен масив – представяне и достъп до елементите

- **Всеки отделен елемент на масив се достъпва** с помощта на **името на променливата** на масива **и** в правоъгълни скоби се записва **индекс** – arr[0], arr[1]…, arr[size-1], където size е променлива, в която сме записали броят на елементите.
- В някои ЕП масивите са сложни обекти, които носят информация не само за елементите, а и за броя им (напр. arr.length). В C++ броят на елементите не може да се получи чрез името на масива.
- При обръщение към несъществуващ индекс в C++ не се получава съобщение за грешка – т.е. може да се променя памет, която е предвидена за други неща (данни и др.).
- Обхождането на елементите на масив в цикъл става като управляващата променлива се променя от 0 до size-1 със стъпка 1.

Инициализация на масив. Дефиниране на масив

- При декларация на масив в C++ стойностите са неопределени.
- Самостоятелно трябва да се задават желаните стойности за елементите на масив:

```
const int size = 3;
int arr[size];
arr[0] = 5;
arr[1] = 6;
arr[2] = 7;
```

- Възможно е да се дефинира масив и инициализират елементите му при декларацията като във фигурни скоби се изброят желаните елементи. В този случай може да не се зададва броя на елементите:

```
int arr2[3] = {2, 3, 4};
int arr3[]  = {2, 3, 4};
string names[] = {"Иван", "Мария", "Петър", "Петя"};
```

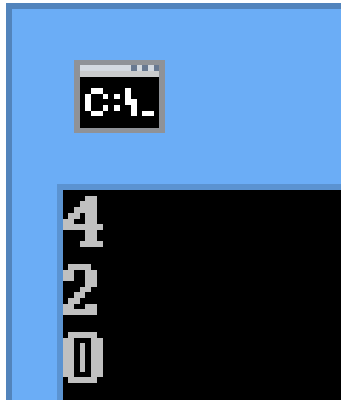
Обхождане на масив в цикъл

В цикъл може да се променя стойността на управляваща променлива, обикновено именувана с i, от 0 до „броя на елементите-1“ със стъпка 1. Тогава, при всяка итерация, в тялото се работи с i-тия елемент.

```
const int size = 3;
int arr[size];
// i се променя от 0 до size-1 със стъпка 1
for (int i = 0; i < size; i++) {
        arr[i] = i * 2; // на i-тия елемент задаваме стойност i*2
}
// последователно отпечатване на елементите на масив
for (int i = 0; i < size; i++) {
        cout << arr[i] << '\n';
}
```

Обхождане на елементите в обратен ред

```
for (int i = size-1; i >= 0; i--) {
     cout << arr[i] << '\n';
}
```



Обхождане на масив с конструкция for-each

- Обхождането се извършва само от първия към последния елемент;
- Използва се само за статични масиви (и други сложни типове, но не и за динамични масиви);
- От версия C++11;
- Използва се ако в програмата индексите на елементите не са от значение.

Синтаксис на for-each

Синтаксис:

for each (<стойност> in <колекция>),

- **колекция** е променлива за колекция (в частност масив), в която се описва множество от елементи;
- **стойност** е променлива за текущия елемент от колекцията
  - Трябва да е от типа на елементите на колекцията.
  - При всяка итерация в нея автоматично се записва стойността на текущия елемент на масива.
  - При първата итерация – стойността на първия елемент, при втората – на втория и т.н.
  - Ако желаем, чрез променливата да променяме съответния елемент в масива/колекцията, то пред името на тази променлива задаваме знака & (за псевдоним – за които ще говорим в следваща лекция).

Съкратен синтаксис

for (<стойност> : <колекция>)

Обхождане с цикъл for-each - пример

```
// За всеки елемент на arr, реферирам от el_value...
for each(int &el_value in arr) {
     cout << el_value << '\n';
}

for(int &el_value : arr) {
     cout << el_value << '\n';
}
```

Генериране на масив от (псевдо) случайни числа

(без разяснения за функциите за генериране на случайни числа)

```
#include <iostream>
#include <string>
#include <ctime>   // системно време
using namespace std;


int main(){
    const int size = 3;
    int arr[size];

     // инициализация на генератора на псеводослучайни числа с текущото време
    srand(time(0));
    for (int &el_value : arr) {
        el_value = rand(); // генериране на псевдослучайно число
        cout << el_value << '\n';
    }
    return 0;
}
```

Функция за връщане на масив като низ

```
// Връща масив като низ във вид {el_1, el_2,...el_n}
// Параметри: масив arr и брой на елементите size.
string arrayToString(int arr[], int size) {
    string s = "{";

    for (int i = 0; i < size; i++) {
        s += to_string(arr[i]) + (i<size-1?", ":"");
        // добавя запетая само ако не е достигнат последния елемент
    }

    s += "}";

    return s;
}
```

Генериране и отпечатване на масив

```
#include <iostream>
#include <string>
#include <ctime>
using namespace std;
```

```
int main(){
    const int size = 5;
    int arr[size];

    srand(time(0));
    for (int& el_value : arr) {
        el_value = rand();
    }

    cout << "Array: " << arrayToString(arr, size) << '\n';

    return 0;
}
```